Everything is a file in Linux, including the commands that you have been using. In this activity I will prove it to you.

1. Navigate to the `/` (root) directory and take a look at what is inside it

```
$ cd /
$ ls
```

2. You should see all the directories we just discussed and possibly more which we didn't cover. In this activity we are going to focus on the `/bin` and `/usr/bin` directories. Let's take a look at `/bin` first:

```
$ cd /bin
$ ls
```

3. You should be seeing a lot of stuff on your screen but among them are names you are very familiar with like `ls`, `cd`, `cat` etc. In fact, all the commands we've covered so far are in this directory. As you can see all these commands are files. Let's see what is inside the file for ls:

```
$ cat ls
```

4. If you see a bunch of garbage get dumped out on the screen then yes you typed the command correctly. Even though these commands are files they do not contain content that is in a human readable format. They are binary files meaning that they were once (probably) C code but got compiled and packaged into a binary file that has machine code which your computer understands. If you actually want to look at the C code for a linux command it is publicly available online. Ok now back to the good stuff, let's view the contents of the cat binary file by running:

```
$ cat cat
```

5. Now that we are familiar with `/bin` let's see what `/usr/bin` is all about:

```
$ cd /usr/bin
$ ls
```

6. Wait what?? This looks exactly the same as `/bin` . I can even `cat cat` like before. These directories do In fact have the exact same content. If you want

to convince yourself of this you can use the cmp command to compare the two files for ls or any other command:

```
$ cmp /bin/ls /usr/bin/ls
```

Info about the cmp command can be found here. TLDR if you don't get any output after running this command it means that the files have identical contents ;)

7. So the question is, when I use the `ls` command, which one am I using?

Luckily we have a command to help us figure this out! Execute the following:

```
$ which ls
```

8. Now we can see that `/usr/bin/ls` is the real `ls` and `/bin/ls` seems to be an imposter. Well, this is not actually true. There is only one copy of the ls binary file that exist on the system. If you run the following command:

```
$ ls -l /
```

You will observe that the `/bin` directory is shown as `bin -> usr/bin` which means that `/bin` is a symbolic link to the `/usr/bin` directory. Ok so what does that actually mean? It means that `/bin` doesn't actually have anything inside of it, it simply points to the `/usr/bin` directory contents when you look inside it. You can think of `/usr/bin` as a room inside of a house. You open the door and there is a desk, a bed, personal belongings, etc. You can think of `/bin` as just a door that when you open it, it has a portal that allows you to walk into the room that is behind the `/usr/bin` door. Everything in that room is only stored in one place but you can access it using both doors.